Imaginary Soundscapes: the SoDA Project

Matteo Casu
Celi srl
Via San Quintino 31
Turin, Italy
casu@celi.it

Marinos Koutsomichalis
CIRMA/StudiUm
Università di Torino
Turin, Italy
marinos.koutsomichalis
@unito.it

Andrea Valle
CIRMA/StudiUm
Università di Torino
Turin, Italy
andrea.valle@unito.it

ABSTRACT

The SoDA (Sound Design Accelerator) project aims at providing a flexible software environment for soundscape generation. Based on semantic information, it provides both an annotation schema and an annotated library of sound files that operates in relation to a generative system that delivers the final audio content. SoDA provides the user with various forms of interaction: from incremental assisted exploration of semantic and audio content in real-time to completely automated off-line soundscape composition. In this paper we describe the semantic and audio components and the various interaction modes.

Categories and Subject Descriptors

H.5.5 [**Sound and Music Computing**]: Modeling

General Terms

Human Factors, Standardization

Keywords

Sound Design, Soundscape, Ontologies, Information Retrieval

1. INTRODUCTION

Sound design is now a term encompassing various domains and applications, from non-interactive audiovisual products to real-time scenarios including virtual reality, multimedia installations, soundscape generation for urban and architectural design. As a consequence, the sound designer has to deal with very different tasks and contexts. But indeed a very general requirement s/he has to deal with is the production of soundscapes, here intended broadly as background sounds that includes the so-called "ambiences" in the audiovisual domain and "atmospheres" in other contexts, intended as continuous but possibly varying sound layers. As shown since the foundational studies by Murray Schafer [12], a soundscape is not a technical artefact but a semiotic one, and its description must take into account semantic (i.e., broadly speaking, "cultural") aspects. From the point of view of practices [7], sound design relies on a hybrid combination of high-level, abstract skills –that properly belong to design itself– and lower-level, more technologically oriented ones, as the designer has to deal also with the production phase (that is, producing the actual sound content). Thus, while dealing with soundscape production, the sound designer has to mediate between the conceptual level and its technical realisation, as s/he is in charge of both. A possible drawback is that the technological side might be overwhelming, as the final output is typically a part of a more complex product, that has to be delivered in relation to a certain commission, thus leaving little room for conceptualisation. On the other side, the actual work of the sound designer is based on a constant, crucial feedback between design decisions and sound results. While a software cannot substitute human skills in designing the foreground elements of a scene (e.g., the lead sound elements of a scene in an action movie), it is thus interesting to tackle the issue of a (semi-)automated soundscape generation that takes into account the semiotic level, i.e. that would be triggered by descriptions such as "people chatting on the streets", "dogs barking on the beach", "a forest with animal calls", "horns of cars from the eighties", and the like. The SoDA project is aimed at providing such a possibility to the user by coupling a semantic component, that includes advanced search features together with faceted filtering and navigation to specify a soundscape, with a sound synthesis component, that includes both a soundscape generator and a soundscape composer.

AM '14 October 01 - 03 2014, Aalborg, Denmark

ACM ACM 978-1-4503-3032-9/14/10 ...$15.00.
http://dx.doi.org/10.1145/2636879.2636885

2. AN OVERVIEW OF SODA

Figure 1 (left) provides a general formalisation of standard sound design practices in the audiovisual domain, which is still the most relevant (see in general [7]). First, the sound designer retrieves sound files from an already existing archive (being it created from scratch or commercially available). Then, the resulting sound files have to be organised following a certain schema (e.g. layering). Finally the tracks have to be processed and mixed so that the final audio is available. Taking into account the previously discussed issues, SoDA aims at providing –with respect to the creation of soundscapes– a twofold computational "acceleration" (hence its name) to sound design practice: on the selection of relevant sound elements to be composited and on their organisation. The automated modules provided by SoDA are shown in Figure 1 (right). In SoDA sound files have been annotated partially by human experts with tags preserving relevant information (see later) partially by

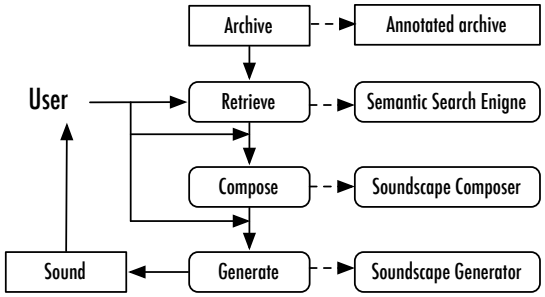

Figure 1: General formalisation of sound design for soundscapes (left) and automatisation in SoDA(right).

software procedures that annotate the files with technical metadata extracted from the file (administrative metadata *and* spectral information). The files are then stored into an archive. The capability of searching and retrieving the files is provided by a Semantic Search Engine module, that allows to retrieve sound files on the basis of inputs (queries and selections) performed by the user. On the other side, SoDA features an automated Soundscape Composer that allows a rapid, tuneable, creation of soundscapes of unspecified duration. The Soundscape Composer receives the results from the Semantic Search Engine and is provided with algorithms for semantically-informed, automated layering. Finally these data –placed at the organisational level– are passed to the Soundscape Generator which is responsible for the final audio rendering. Soundscape Generator relies on sound-localisation techniques as well as to filters, reverb-units and digital resonators in order to synthesise a three-dimensional soundscape.
In the following we will first discuss the semantic annotation schema and the semantic engine for retrieving sound files, and then the two components that implement the strictly audio related parts.

3. ANNOTATION AND SEMANTIC SEARCH

In order to enhance semantically the retrieval of sound files, we have defined an annotation schema starting from the way in which metadata are annotated in state-of-the-art sound libraries.
Examples of widely diffused libraries on the market are Hollywood Edge[1], SoundIdeas [2], Blastwave FX [3]. Among the libraries used by sound designers we have taken into consideration Sound Ideas Series (6000, 7000 and 10000) and World Series of Sound[4], and Renaissance SFX[5]. The search tools for audio documents taken into account are SoundMiner[6], Library Monkey[7], Basehead[8], Audiofinder[9], Apple iTunes (that, even if not intended for professional use, is used by small studios). An example is the metadata tagset from Renaissance SFX, that includes: identifier, title, length, type of ambience, position (stationary or echo), movement (direction of the sound), description, category (with respect to an internal classification). Regarding how documents are usually tagged in commercial sound libraries two observations can be made:

- the "description" field often seems to include information that could better be served in dedicated fields. A typical example is the information about the location represented in the sound file;
- libraries rarely follow standard and controlled vocabularies: each library uses a specific structure for the fields of its documents and for their values. This is a problem in terms of interoperability between libraries as well as in terms of ease of use for a user (in our case, a sound designer). An example of this inconsistency is given by the various formats used for dates, or by different ways in which the genre is tagged.

As for administrative and technical metadata, the Audio Engineering Society (AES) has published two standards[10]: AES57-2011 ("Audio object structures for preservation and restoration") and AES60-2011 ("Core audio metadata"). There are also less specific (but freely available) standards, such as the Dublin Core, in its variants ("application profiles"[11]). Other, more domain-specific, examples, are EBU-core (AES60) and MPEG-7 (including the audio section ISO/IEC 15938-4:2002). In 2012 the WWW consortium (W3C) mapped some of the most used schemas for media objects in the Ontology for Media Resources[12], recommending it for the annotation of digital media on the web. The W3C specification is not bound to a serialisation in a particular language, so it can be used as a general schema for the SoDA project.
Within SoDA, the Semantic Search Engine heavily relies on classical techniques borrowed from Information Retrieval (IR) [10], whose main task is finding relevant "documents" on the basis of the user's information needs, expressed to the system by a query. The process of feeding the search system with documents has to be preceded by the task of annotating documents (files) with metadata, which are then used by the system to provide features useful to the user, such as faceted search (i.e., the capability of filtering documents by selecting orthogonal features) and query expansion (a technique in which the engine recognises relevant concepts in the user query and triggers an "expanded query" on the documents). The "intelligence" shown by the system depends on several factors. The two most relevant ones are: a good annotation of the documents and the availability of a knowledge base that helps the system in recognising concepts within the metadata and in the user's query (so that the search is *mediated* by the knowledge base).
Obviously, the process of annotation imposes trade-offs between scalability and accuracy: a structured manual annotation, which uses a tagset compliant to the knowledge base used by the system, will likely lead to better IR results, but it will hardly scale to situations in which new documents are

[1]http://www.hollywoodedge.com/
[2]http://www.sound-ideas.com
[3]http://www.blastwavefx.com/index.html
[4]http://www.sound-ideas.com
[5]http://www.renaissancesfx.com
[6]http://store.soundminer.com
[7]http://www.monkey-tools.com/products/library-monkey/
[8]http://www.baseheadinc.com
[9]http://www.icedaudio.com
[10]http://www.aes.org/publications/standards/
[11]http://dublincore.org/documents/profile-guidelines/
[12]http://www.w3.org/TR/mediaont-10/

created and must be annotated by hand. A good compromise is hence to conceive the document as a set of different fields, some of which can be partially structured (e.g. contain free text, which is easy for a human to manage) and others can be pre-compiled by means of an automated process. Therefore, choosing the right tagset for the annotation is quite a crucial task: the tagset should cover the project needs but also be kept simple, in order to be usable. Moreover, it is a good choice to keep it interoperable with existing standards. In SoDA metadata are divided into three sets:

- *administrative metadata*: provenance information, copyright, date of creation, etc. ;
- *technical metadata*: file format, length, number of channels, and spectral information;
- *content metadata*: data about the content described by the document;

It is important for SoDA to have a scalable production model: how much effort is required to add a new set of sound files to the system? To this regard, administrative metadata are typically created together with the creation of the audio file, while scalability on technical metadata is guaranteed by a custom audio analysis utility that processes the library in batch mode and pre-compiles technical metadata on each document. Only content metadata must be compiled by a human; it is not mandatory for the human to compile all the content fields – it is also possible to provide unstructured text. In this case, the search engine will have less structured information available, but the search will still operate on the content field.
In Figure 2 we give the set of tags used in SoDA by means of an example. Our tagset takes into account both standard practices and the needs explicitly expressed by the sound designers involved in the project. Among these are the number of channels, the location, and information about the context: season, time of the day, meteo. We also tried to stay as compliant as possible to the *Ontology for Media Resources* as for the names and types of the fields. Among content data, the tag `typeOfSoundObject` is related to a phenomenological appreciation [15], and allow to define Atmospheres (long sounds, with no begin/end), Sound Objects (atomic sounds), Sequences (composite sounds that nevertheless show a unique identity). This 3-element typology is relevant for soundscape composition (see later, an analogous classification is proposed in [16]). Some of the content data cannot be properly represented by using a flat tagset, but needs some structure: as an example, locations constitute a taxonomy (New York is a place in the United States, which are part of North America). Physical objects, present together with the locations in the text descriptions, generally have a similar structure: e.g., birds are animals. The SoDA Semantic Search Engine uses as knowledge base an OWL artifact [13][13].
The OWL ontology is constituted by two main taxonomies: locations and objects, among which there are physical as well as abstract objects (including some events). The hierarchical relations among objects and among locations are

[13]Using a widespread terminological abuse, we here use the terms "knowledge base" and "ontology" interchangeably, and use the term "concept" to refer to a class or an individual.

```
<doc>
    <!– management metadata –>
  <field name="identifier">1234</field>
  <field name="creator">Machiavelli Music</field>
  <field name="collectionName">My Great Collection</field>
  <field name="copyright">2013 Machiavelli International Musical Images - All rights reserved</field>
  <field name="recordDate">2013-11-12</field>
  <field name="releaseDate">2013-11-15</field>
  <field name="description">dogs barking loud in the streets of turin</field>
  <field name="title">dogs barking</field>
    <field name="url">http://fakedomain.com/docs/1234</field>
    <!– technical metadata –>
  <field name="bitDepth">24</field>
  <field name="channels">6</field>
  <field name="duration">00:00:12</field>
  <field name="samplingRate">192000</field>
  <field name="typeOfShot">closeup</field>          <!– closeup, mid-shot, long-shot –>
    <!– technical metadata –>
  <field name="centroid">520</field>                <!– in hertz –>
  <field name="complexity">0.8</field>
  <field name="dissonance">0.5</field>
  <field name="loudness">3.2</field>                <!– in sones –>
  <field name="onsets">3.2</field>                  <!– array of seconds –>
  <field name="onsets">2.5</field>
  <field name="onsets">1.9</field>
  <field name="pitch">100</field>                   <!– hertz –>
  <field name="sharpness">0.3</field>
  <field name="slope">12</field>
  <field name="spread">1</field>
  <field name="weightedSpectralMaximum">100</field>
    <!– content metadata –>
  <field name="createdIn">torino</field>            <!– place of the recording –>
  <field name="depictsFictionalLocation"></field>   <!– place depicted by the recording –>
  <field name="season"></field>                     <!– spring, summer, autumn or winter –>
  <field name="timeOfTheDay"></field>               <!– morning, noon, evening, night –>
  <field name="content">dogs</field>                <!– content associated to the file (e.g. the source of the sound) –>
  <field name="periodStartYear"></field>
  <field name="periodEndYear"></field>
  <field name="typeOfSoundObject">atoms</field>         <!– sound object, sequence, atmos–>
</doc>
```

Figure 2: An example of SoDA document.

used by the search engine to perform query expansion; sets of synonyms attached to each individual or class are also used by the engine to retrieve the concepts in the text. The total number of individuals in the ontology is roughly 1000 (≈300 locations and ≈700 objects). The ontology has been built with a bootstrap process: a set of locations has been extracted from GeoNames (in particular, countries of the world with their major cities), while the part on physical objects is a custom-made extension of the Proton Ontology, an Upper Level Ontology by OntoText[14].
A set of experimental studies about how people classify sounds are reviewed in [6]. The authors distinguish between different types of sound similarities: acoustical similarity, event similarity (based on the physical similarity of the events that cause the sound), semantic similarity (based on the knowledge associated to the identified object causing or related to the sound). From the point of view of these distinctions, since Soda interacts with the user by means of textual queries, the semantic dimension is the prevailing one while retrieval of sound objects. The acoustical dimension will be included in future releases including psychoacoustic and fenomenological classes: we plan to introduce in the Soda ontology a set of classes defined in terms of spectral information: on one side, psychoacoustic classes (such as the one by Gaver[3], who proposed a typology of sounds in relation to their perceived acoustic production), on another side experimental phenomenological classes (such as "calm" or "noisy"), in order to translate the user's queries into structured queries over the documents. This feature

[14]http://www.ontotext.com/proton-ontology

will be developed in later works, with the aim of providing the sound designer with an innovative tool to explore sound archives. Indeed, the possible discrepancy between the perceived sound and its real origin (e.g. a closing door might not "sound" as a closing door, rather as something different that nonetheless might be used appropriately in a completely different contexts) has been widely discussed since the beginning of *musique concrète* [15].
Thanks to the linked data links between GeoNames and DBPedia [5] demonyms are also used by the engine: "french" finds "France" (and children locations) and vice versa. The ontology has also been manually enriched with correlations between classes (for example, between "city" and "cars") which are used to provide the user with suggestions about related content. Part of the correlations are automatically added by an automated reasoner. In order to help the Composer module to generate sensible soundscapes, the ontology also keeps track of information about the type of sound emitted by the source, e.g. if the sound is repeatable.
Thanks to this data organisation, together with basic linguistic analysis (tokenisation and lemmatisation) the Semantic Search Engine permits to find children as well as syntactical variants of the concepts expressed in the user query, gaining a simple yet effective set of IR features.
Regarding the scalability issue, as we have seen the ontology largely re-uses existing knowledge bases, while the part annotating sound features is human-made, even if the effort is performed once for each update of the ontology.

4. SEMANTIC DISCOVERY

In the previous section we have discussed the annotation schema. The resulting annotated library can be indeed explored interactively by the sound designer while looking for certain sounds. Moreover, in the SoDA project, the library and the search engine are parts of the system leading from semantic search to soundscape composition. The search engine exploits the ontology during the search (for query expansion). Furthermore it permits to filter the documents along the different dimensions annotated on the documents. So, thanks to the rich annotations (together with the matches between the documents and the ontology), the user is provided with the ability to compose different flavours of search: administrative/technical (e.g. filtering by collection or type of shot), content-based (places, objects, season), and temporal (filtering on the time of the day or by decade). The filters can be combined by using boolean operators, so that it is very simple to select, e.g. documents of medium shot annotated with the *winter* season and depicting a place which is not in the UK.
It is easy to see that the engine permits two main search modalities, that can be mixed: a Google-like search in a text box and the selection of "facets". Facets are also useful in providing the user with a clue about the content of the documents. Given the features exposed, an example of search which combines them can be described by the following scenario: suppose the user search for "cars in the nineties" by filling the input form with the very sentence. The engine recognises the concepts of *car* and *nineties*. The search is hence routed to the terms "car" and "cars", together with the terms associated to all the children of the concept which are in the ontology (e.g. "Jaguar"). As for the concept of *nineties*, the engine routes this part of the search to a range query over the fields *periodStartDate* and *periodEndDate*.
The suggestion of classes related to *car* (such as *city*) can be used by the user to expand its original query or to trigger a new one. These two modalities can be better described by considering again a real scenario: when the user search for "streets of new york" the engine retrieve sound files by using the concepts recognised in the query: in this case, *street* and *new york* (if part of the query is not recognised, it is still used as text). The retrieved sound files will contain the two concepts. However, in some cases this will not be sufficient to generate what the user likely expects: for example, not all files with sound of streets will contain *car* or *chatting*. Now, suppose that the user has searched for "streets of new york" but has one of the two following use cases in mind:

- the user needs (sounds of) streets of New York with cars and people noises
- the user needs the streets of New York without those two concepts

This is where the suggestion of correlations comes at rescue: the system propose the two concepts *car* and *chatting* as concepts related to the query. The user can now decide to add them to the original query, in AND for use case 1 or in NOT for use case 2.
At the price of further annotation in the ontology, the system can recognise a query like "streets of new york - calm" as excluding loud things like car horns. The suggestion of correlations has another advantage, which is the serendipity: the user can be proposed with useful content which he/she had not considered at first. Figure 3 show a search result

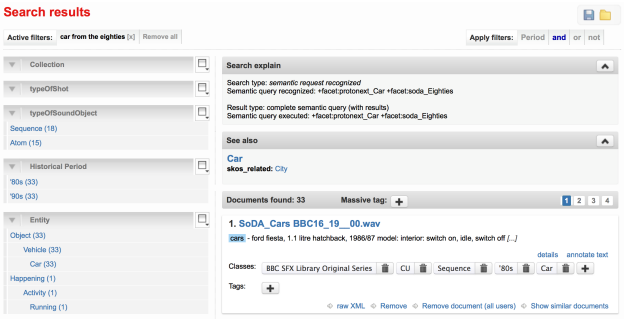

Figure 3: An example of the results provided in response to user search.

for the query "cars from the eighties". The facet trees show the facets which are active for the documents found, with the number of documents matched against each facet, while the "See also" section suggests for concepts related to the ones recognised in the query. The "Search explain" section provides a feedback about the recognised concepts. Figure 4 shows the details of a document in the search engine. The example represents a document before the spectral properties are added by the automated processing utility.
While in the case of a human user these kinds of interactions can be exploited in an interactive fashion, in the case of a software client querying the Semantic Engine the default behaviours of the client must follow predefined heuristics. This is the case of the Soundscape Composer, which is described in following sections.

1. SoDA_Cars BBC16_19__00.wav

Administrative metadata

identifier:	soda_cars_bbc16_19__00.wav
creator:	
collectionName:	BBC SFX Library Original Series
copyright:	
recordDate:	
releaseDate:	
title:	SoDA_Cars BBC16_19__00.wav
url:	ftp://91.212.167.101/Editing/140306/

Technical metadata

bitDepth:	
channels:	
duration:	
samplingRate:	
typeOfShot:	CU

Content metadata

createdIn:	
depictsFictionalLocation:	
season:	
timeOfTheDay:	
description:	CARS - FORD FIESTA, 1.1 LITRE HATCHBACK, 1986/87 MODEL: INTERIOR: SWITCH ON, IDLE, SWITCH OFF
periodStartYear:	1986
periodEndYear:	1995

Figure 4: An example of the results provided in response to user search.

5. MODELLING IMAGINARY SOUNDSCAPES

SoDA's design involves Soundscape Generator (hence on SSG), a soundscape synthesis engine capable of modelling three-dimensional soundscapes with a variable degree of realism. SSG is designed as a versatile and multi-featured autonomous audio engine, that can be integrated modularly into SoDA. SSG is capable of modelling complex 3D spaces and of providing localisation of sound events within them and with respect to their acoustic properties. SSG works both in real and non-real time, in order to be used for both quick tests and experimentation as well as for its specialised role or in SoDA. It delivers sound in a variety of formats (mono, stereo, multichannel, etc). Finally, it can emulate and re-construct complex soundscapes by means of minimal sonic material.

With respect to similar systems ([17], [11], [2],[16]), SSG addresses all the aforementioned issues and maintains a modular nature. This means that it can be exploited at different user level: as a low-level audio engine or by means high-level abstractions such as the ones provided by the Soundscape Composer (see later), that hide SSG to the user allowing her/him to focus on soundscape organisation from a more semantic-oriented perspective. SSG generates audio by taking into account three elements: a "Space", a "Listener", and a "Decoder" (see Figure 5).

The "Space" is intended as a model of the desired space. A Space is built as an aggregation of an arbitrary number of individual "Zones" and with respect to their individual geographic, acoustic and sonic characteristics. Zone's

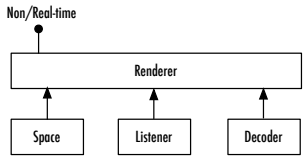

Figure 5: Structure of SSG.

geographical features refer to their spatial boundaries and their absolute positioning in a 3D virtual space –both are to be specified using three-dimensional Cartesian coordinates. Their acoustic features refer to modelled physical phenomena such as reverberation, resonance, acoustic absorption, and others, and are to be specified as arrays of parameters passed to the synthesis procedures. Their sonic features refer to the type of sound events that may occur inside a Zone and to be specified using an arbitrary number of individual "Source" objects.

All Sources are conceived as containers for some sort of audio event which may or may not be repeated in time –their only differences lying in their spatial positioning and directionality. That is, a Source consists of both the audio data to be reproduced as well as the information regarding the *when* and *where* they should be reproduced. The *when* is defined by means of a pattern-based mechanism (see next section), while the *where* by virtue of selecting the appropriate kind of Source and parametrising it accordingly. SSG implements five different Sources: an Atmosphere (non-directional background sound), a Fixed-Sound (directional and fixed in space), two types of Ambulatory-Sounds (governed by envelopes or by a user-definable behaviour, respectively) and a Sound-Cloud, representing complex events, such as e.g. rain or crowds, that are characterised by multiple appearances of similar sonic sequences at ever-changing and random positions within a given volume and with respect to a density factor.

"Listeners" are intended as models for the navigation through the soundscape as defined in the Space, in order to offer an anthropocentric listening perspective on it. SSG provides the sound designer various kinds of specialised Listeners that share some general features. These include Listeners that are fixed in space or that are movable, as well as a "Random Walker" which will perform three-dimensional random walks on a given volume. Two types of movable Listeners exist, allowing their movement in space to be controlled by either three envelope functions (one for each spatial dimension) or using some user definable callback function that returns new Cartesian coordinates on regular intervals. The ability to differentiate the properties of each individual Zone, together with the ability to have movable Listeners makes it theoretically possible to design imaginary sound-walks through three-dimensional soundscapes. For instance, a walk through the miscellaneous rooms of an imaginary skyscraper or through a cityscape and inside various buildings or other constructs.

Finally, the "Decoder" manages the desired output format. Internally, SSG relies on ambisonics spatialisation algorithms [9] so that, given the opportune decoder, the same audio stream may be decoded to one of the standard formats such as mono, stereo, 5.1, 7.1 or even for reproduction from an arbitrarily configuration of speakers in 2D or 3D space (even

if canonical setup give indeed better results).
The user (be it human or machine) thus designs an imaginary soundscape by providing SSG's main "Renderer" the three necessary elements (Space, Listener, Decoder). The Renderer is able on request to directly stream audio (real-time mode) or to bounce it to an audio file (non-real time mode).

6. SOURCES AND PATTERNS

One of the most difficult aspects in soundscape simulation and generation concerns the modellisation of source behaviour in time. A precise physical modelling would not be possible for a general framework like SoDA as it would require an encyclopaedic knowledge base in order to be able to correctly describe a very large set of possible sound sources. Moreover, SoDA is based on sound samples as stored in the library, adhering to a typical working pipeline of sound designers. Thus, its operation mode is necessarily related to playback (even if samples may be processed, e.g. by implementing space-related processing like reverb etc). An interesting approach in relation to sound synthesis is "cartoonification" as devised by the Sounding Object project [14]: here sounds are described not in terms of the real mechanics of their production, but following a phenomenologically-compliant, physically-simplified, ecologically-related approach, starting from Gaver's studies that we have mentioned above. Cartoonification can then be extended from single sounds to sound sequences. In this case, cartoonification can be considered as a phenomenological description of time-organisation of sound events that does not attempt at defining analytically the mechanics of the sound source. Rather, it aims at providing a basic "rhythmical" description that tries to match the perceptual organisation. In SoDA, time-organisation of sound events is thus cartoonified (loosely modelled) by means of specific data structures, generators [1]. A generator may be thought as a rule for generating sequences, much in the sense in which a vectorial representation for graphics is a rule for generating the actual image as opposite to exhaustively storing all the the data in a raster image. When executed, a generator will result in a stream of values of a certain length and with respect to some high-level rule. As the rule is specified rather than the actual data, the sequence can be of infinite length. In relation to Sources, a generator-based strategy allows for quick and efficient emulation of a variety of time behaviours: from one-shot to repetitive sound-events, from deterministic to stochastic sequences. Thus, in SSG, Sources have to be associated with some generator that defines the exact time of their first appearance and their repetition scheme. Support for generators is native in the SuperCollider language (used to implement SSG) by means of "Patterns" and "Streams" of data that result from their execution [4]. Thus, hence on we will use the term "pattern" and the relative SuperCollider notation. Built-in patterns include representations for linear sequences, random selections from lists, probability-based number generators, random walks, and other similar mathematical constructs that provide a conceptually straightforward way to model streams of values. Such patterns may be chained and/or nested recursively, allowing for a very compact notation of complex behaviours. Consider, for instance the following pattern:

$$Pseq([0,1],2)$$

It is a pattern ("P") defining a linear sequence ("seq") of 0 and 1 repeated twice (2), representing time intervals. As a model of temporal behaviour, it will result in the Source being reproduced 4 times: immediately once the Renderer is asked to play (0 seconds); 1 second after having finished playing back; immediately after this second appearance has finished (again, 0 seconds); and 1 second after the third appearance. While such a behaviour might be easy to model otherwise, consider how simple it is for a user to model highly complex behaviour with a minimal notation as in the following example:

$$Pseq([0, Pexprand(0,10,1), Pwrand([0,4,2],[0.6,0.2,0.2],2)],3)$$

This pattern is recursively defined with other patterns. It will result in the Source being reproduced immediately; then (once having finished) after a random number of seconds selected from an exponential numerical distribution between 0 and 10 ("Pexprand"); then again after 0, 4 or 2 seconds and with respect to the probability table [60%,20%,20%] for each number, respectively. The last pattern is repeated twice, then the whole cycle is repeated 3 times. Therefore the total numbers of repeats will be 12 and will follow a specific, albeit non-deterministic, behaviour. Indeed, rather than random permutations of sound sources, real-life soundscapes present specific spatial and temporal patterns with a a variable degree of randomness. Complex temporal patterns like this have been found by zoomusicological studies e.g. in bird singing [8]. Thus, from the user's perspective, patterns provide a compact notation for realistic modelling of soundscape layers, by describing the phenomenological organisation that results from their physical behaviour, indifferently from their natural or artificial origin (e.g. birds, rain, cars). As an example, a dog barks in an irregular but not random way. As the annotated SoDA library keeps in the database recordings of barking sequences, it is easy to model a highly realistic, ever-permuting dog barking by means of patterns that describe the phenomenological properties of a dog barking applied to a very small set of bark samples. Patterns allow the user to model this mechanism with ease and with respect to a more natural way of thinking about sound events. For instance, if a dog's barking pattern is understood as if occurring every now and then in irregular patterns and having a varying duration, then a barking sequence maybe easily represented as follows:

$$Prand([a,b,s],inf)$$

$$Pwhite(0.5,2.0,rrand(3,10))$$

The first Pattern defines the atomic sounds that will be used, with a and b representing two different bark atoms, s standing for silence and inf for infinity (it is upon the duration pattern to define the total number of atoms used). The second pattern defines their durations as random numbers between 0.5 and 2 seconds and will aggregate a random number of atoms between 3 and 10 (the $rrand$ function). Then, a Source which points at the aforementioned sequence will cause it to generate a new contingent audio sequence whenever needed and with respect to this particular repetition schemata (as already demonstrated). The result will be a realistic dog barking, each repetition of which being different, yet, identifiable as belonging to the same dog. A graphical representation of a possible barking sequence (here made up of 7 sounds for ≈ 9 seconds) is shown in Figure 6,

where each bark sound is given an index (1 and 2) and silence is represented by 0. Patterns can be easily tuned to

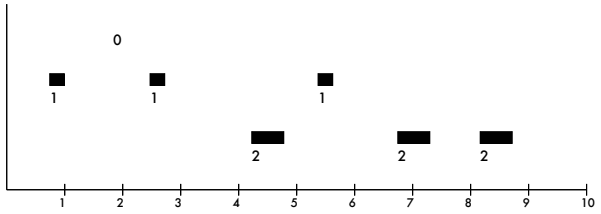

Figure 6: A barking sequence.

other time organisations, such as the Markov chains in a graph formalism that has already been proposed to represent soundscape layers (e.g. human activities such as selling goods in a market) [16]. As demonstrated in the previous example, employing a pattern-based strategy is a powerful and accurate sequencing mechanism. More, it exhibits a more anthropocentric kind of logic, since it is based on simple and conceptually straightforward representations, and, therefore, allow the user to interact with SSG in an simple and efficient way.

7. SOUNDSCAPE COMPOSITION

While it is possible to model complex soundscapes directly with SSG, SoDA asks for a fully automated soundscape composing paradigm. Thus, a Soundscape Composer (SSC) module has being conceived as a bridge between SSG and the other components of SoDA. Unlike SSG, which is conceived as an autonomous module, SSC is a SoDA-specific construct whose primary goal is to restrict and specialise SSG according to SoDA requirements.
SSC has several tasks to address, in particular:

1. parse and interpret the results of the semantical analysis engine;
2. model a space with the adequate geographical and acoustic features;
3. populate it with sources;
4. finally, place a listener within it

In the context of SoDA, a soundscape is intended as a background ambience with some possible moving sources: thus, a Space is a singleton Zone inhabited by a fixed Listener. To attain this goal, SSC imposes many higher-level constraints in relation to the many low-level control possibilities provided by SSG. First, SSC associates to the SonicSpace a background sound that keeps on looping constantly, in order to provide a first background layer. Then, SSC generates individual sources and sequences. To accomplish such a task, SSC relies on three modules: an algorithmically generated behaviour list; a rule-based engine, featuring general rules, stochastic/probability functions and user-defined regulations; a memory manager (Figure 7).

The behaviour list is intended as a description of the way a source behaves. It is built from the ontological data returned by the semantical engine, and provided with default

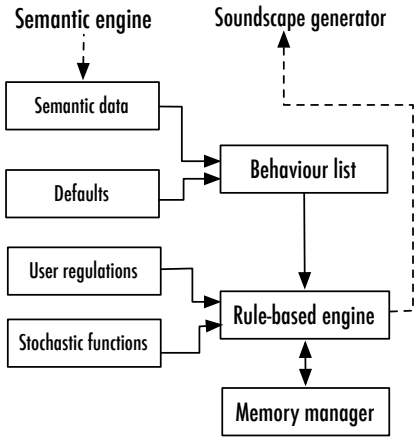

Figure 7: Structure of SSC..

states for a simple everyday typology. For instance, animals and cars move occasionally while buildings do not, speech is generally not to be repeated while animal sounds should repeat but not in a periodic way, cars move mostly in the horizontal level while elevators vertically, etc. A behaviour list consists of straightforward numerical answers to certain properties, such as: the minimum/maximum allowed deviation in each spatial dimension for the localisation of a sonic source, the minimum/maximum speed of their movement and their acceleration pattern, the maximum number of repetitions allowed for a sonic-source, etc.
The rule-based engine is deputed to convert the data provided by the behaviour list into the required Sources and to provide them with their spatio-temporal position and movement features. It takes into account a variety of features, including the spectral ones that have been associated to each sound in the analysis phase of annotation.
The memory manager stores all the data related to the generation: it incrementally populates the memory (initially empty) at runtime with the specifics of the generated sources. Then, and unless the behaviour lists or the user-defined regulations suggest otherwise, SSC attempts to increase variance by consulting the memory manager in order not to generate sequences of objects with almost identical features.
SSG is intended a low-level, fine-tuneable engine. Nevertheless, as we have seen, its control already benefits from the high level formalisation provided by patterns. The purpose of SSC is to define an even higher level, by hiding to the user the complexities involved in manually designing a soundscape. Interaction is intended to happen primarily through the various semantical filters s/he may apply to the query: the semantic search engine becomes the main user interface. Nevertheless, the user may interact directly with SSC by defining a number of parameters. These include technical constraints such as the format of the audio-output (mono/stereo, 5.1, etc) or its total duration. SSC also allows some minimal interaction with its internal intelligence module. The latter is made up of a series of pre-defined behaviours that control the spatial spread of the imaginary soundscape, the approximate density of simultaneous SonicEvents per unit of time, and whether contextually similar sounds should be grouped together or not. By setting these

parameters, the user is able to represent the desired soundscape's overall appearance.

8. CONCLUSIONS AND FUTURE WORK

By coupling semantic annotation and automatic generation, SoDA aims at providing the sound designer a tool for intelligent and fast prototyping, fostering an interactive and explorative approach to sound collection and sound organisation. On the semantic side, SoDA both proposes an annotation schema and provides an annotated library. Through the annotation schema the library can be extended and customised, still allowing the search engine to operate with it. On the other side, the audio components, SSG and SSC, allow for a variable degree of customisation and interaction by the user. The real-time design is focused on providing an immediate aural feedback to the sound designer while s/he is diving into the sound database, enabling her/him to operate with a trial-and-error methodology. This does not mean that a non real-time usage is not possible. On the contrary, an actual implementation of the system is focused on a client/server application, that is, a web service in which the user submits a query to the web interface and gets back a rendered audio file. Such an application is interesting in relation to soundscape, as the production of background sounds is a standard requirement for sound designers, but it is notoriously a time- (and thus resource-) consuming task. The automatic generation of soundscapes is thus relevant for a variety of standard production situations, from fast prototyping in pre-production to low-budget post-productions. Moreover, the reference to a space in SSG allow to automatically obtain a spatial coherence between various sound materials. The generative nature of SoDA is indeed a major point, as it yields always different results, even from the same user query. Moreover, in a real-time situation (e.g. an installation) the generative process ensures an ever-changing soundscape that can be controlled interactively, e.g. by a user associated to a Listener. Evaluation of the results has

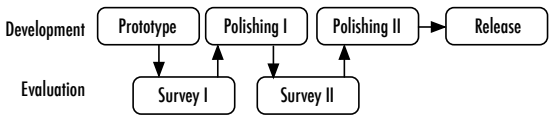

Figure 8: Development/evaluation cycle.

been at the moment conducted in an informal but constant way (as our team includes professional sound designers), but more systematic tests are planned. Such a feedback is indeed very relevant for SoDA, as the project aims at providing the sound designers a tool that can be used in real-world situations. At the moment, we are organising a pool of experts from different professional cultures and backgrounds to implement the procedure shown in Figure 8. Testers will be asked to introduce SoDA in their workflows and to share with us ideas and impressions on the software through an online survey.

9. ACKNOWLEDGMENTS

The SoDA project is founded by Regione Piemonte - *Polo d'Innovazione per la Creatività Digitale e Multimedialità, III programma annuale.* We thank our partners Vittorio Di Tomaso and Andrea Bolioli (Celi), Vito Martinelli and Paolo Armao (Zero dB), and Pietro Giola (Machiavelli).

10. REFERENCES

[1] Budd, T. *A Little Smalltalk*. Addison-Wesley, Reading, Mass., 1987.

[2] Finney, N. Autonomous generation of soundscapes using unstructured sound databases. Master's thesis, Universitat Pompeu Fabra, Barcelona, 2009.

[3] Gaver, W. W. What in the world do we hear? an ecological approach to auditory event perception. *Ecological Pyschology 5*, 1 (1993), 1–29.

[4] Kuivila, R. *The SuperCollider Book*. The MIT Press, Cambridge, MA, 2011, ch. Events and Patterns, pp. 179–205.

[5] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal* (2014).

[6] Lemaitre, G., Houix, O., Visell, Y., Franinović, K., Misdariis, N., and Susini, P. Sound perception, interaction and synthesis. *Deliverable 4.1* (2007).

[7] Lewdall, D. L. *Practical Art of Motion Picture Sound*, 3rd ed. Focal Press, Burlington, MA, 2007.

[8] Mâche, F.-B. *Musique, mythe, nature ou les dauphins d'Arion*. Klincksieck, Paris, 1983.

[9] Malham, D. G., and Myatt, A. 3d sound spatialization using ambisonic techniques. *Computer Music Journal 19*, 4 (1995), 58–70.

[10] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[11] Misra, A., Cook, P. R., and Wang, G. Musical Tapestry: Re-composing Natural Sounds. In *Proceedings of the International Computer Music Conference (ICMC)* (2006).

[12] Murray Schafer, R. *The Tuning of the World*. Knopf, New York, 1977.

[13] OWL Working Group, W. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[14] Rocchesso, D., and Fontana, F., Eds. *The Sounding Object*. Edizioni di Mondo Estremo, Firenze, 2003.

[15] Schaeffer, P. *Traité des objets musicaux*. Seuil, Parigi, 1966.

[16] Valle, A., Lombardo, V., and Schirosa, M. Simulating the soundscape through an analysis/resynthesis methodology. In *Auditory Display*, S. Ystad, M. Aramaki, R. Kronland-Martinet, and K. Jensen, Eds., vol. 5954. Springer, 2010, pp. 330–357.

[17] Warusfel, O., and Eckel, G. LISTEN-Augmenting everyday environments through interactive soundscapes. *Virtual Reality for Public Consumption, IEEE Virtual Reality 2004 Workshop, Chicago IL 27* (2004).